Automatic Speech Recognition (2008-09): Lab-session sheet (2) — Generative models and training algorithms —

Hiroshi Shimodaira (*Revision* : 1.2)

This lab-session is intended to learn Gaussian probability distribution function (pdf) and its mixture, i.e. GMM (Gaussian mixture model), which is widely used for speech recognition (in HTK, of course) and pattern recognition / machine learning.

In the session, you will

- see that GMM can be really employed to generate signals.
- investigate three types of pdfs, (i) histogram, (ii) Gaussian pdf, and (iii) GMM.
- see how model parameters are estimated iteratively by the EM algorithm.
- see the cases when the EM algorithm works and fails.

1 Estimation of pdfs

1.1 Initialisation and invoking Octave

```
% cd ~/asr
% ~hshimoda/pub-asr/bin/init-t2
% cd Octave
% ./octave-gmm
```

Once Octave is up, enter the following command to load an octave program.

```
octave:1> source("gmm-demo.m")
```

1.2 Experiment 1a

Here, we will see a histogram of sampled data generated from a single Gaussian distribution.

```
octave> xd.fname = "gmm-data-m1";                    ··· set parameter file name
octave> GenData                                      ··· generate a data set
octave> PlotHistPdf                  ··· plot a histogram and true pdf of the data
```

- The function GenData() uses a GMM to randomly generate a data set, whose default size is 100. The output is stored into a variable "xn". You can take a look at the data by simply typing the name of the variable, i.e. "xn" at Octave prompt.

  Here, the dimensionality of the data is always one! The GMM parameter file "gmm-data-m1" contains parameters when the number of mixtures is one, i.e. just a single Gaussian pdf.
- The histogram (blue line) shown in your window depicts a normalised histogram of the data set, while the yellow dot-line shows the true probability distribution of the GMM that generated the data set.
- Try GenData() and PlotHistPdf() again by typing "GenData(); PlotHistPdf();". You will see different shapes of histogram for every trial.
- You can change the number of data to generate by giving a first argument to GenData(), i.e. GenData(50) will produce 50 data.
- Increase the number of data to see that the histogram approximates the true pdf.

1.3 Experiment 1b

Now, let's try GMM with two Gaussian distributions. To that end, just change the parameter file by

```
octave> xd.fname = "gmm-data-m2";
```

then repeat the same with Experiment 1a.

1.4 Experiment 2 (training GMM)

```
octave> xd.fname = "gmm-data-m2";
octave> GenData                                  ··· generate a training data set of 100 samples
octave> PlotHistPdf
octave> GmmInit                                          ··· initialise GMM parameters
octave> GmmTrain                                  ··· Train GMM using the EM algorithm
```

- GmmInit() here initialises a GMM with two Gaussian pdfs.
- GmmTrain() will at first show you an initial pdfs of the GMM, and also the same histogram and true pdf of the data as PlotHistPdf() does.
- In the octave dialogue window, you will be prompted to press Return (Enter) key to start the EM training.
- Once the iteration of EM training starts, you will see how the pdfs of the GMM will be updated as the iteration goes on.
- The iteration[1] stops when (i) the change of the optimisation function Q has become less than a predefined threshold, `gmm.th_convergence` (0.1 by default), or (ii) the iteration count has reached a predefined number (100 by default). You can resume iteration by calling GmmTrain() again.

Try the following:

- Change the number of Gaussian pdfs to see how the estimated pdf changes. You can change the number of Gaussian mixtures with setting the first argument GmmInit(*num*), e.g. GmmInit(3) will initialise a GMM with 3 Gaussian pdfs.
- After a trial of GmmTrain(), call GmmInit() and GmmTrain() again to see how initial values of the GMM parameters affect the performance of the EM training in terms of pdf estimation accuracies and convergence speed.
- Try above experiments by using another training data which was generated by another GMM with different parameters. This is done in this way:

```
octave> GenData(100, "gmm-data-m3")
```

1.5 Experiments using real speech data

Instead of using the synthesised data by GMM, you can try real speech data

```
octave> source("lib-htk.m")
octave> ReadMFCC                          ··· store mfcc data to the variable "mfcc".
octave> xn = mfcc(:,1);                                  ··· copy the 1st MFCC into xn.
octave> gc.plot_true_pdf = false;                        ··· suppress drawing true pdf.
octave> PlotHistPdf
```

Details for reading another mfcc data will be announced later.

1.6 Functions

There are two GMMs used in this program, one is for generating training data, and the other is for estimating pdf. To distinguish them, "GMM(gen)" denotes the former.

GenData([*n*, [*filename*]]**):** Generate a data set whose size is n (100 by default). Here the dimensionality of the data is fixed to one. Data is randomly generated using a GMM(gen) whose parameters are read from the file *filename* (default: "gmm-data-m2"). The data is stored in a vector variable "xn".

[1] If you want to accelerate each iteration, change the value of "gc.sleep" to zero, i.e. "gc.sleep = 0;".

PlotHistPdf([*nbins*]): Plot a normalised histogram of the data, and a true pdf of data. *nbins* is the number of bins of the histogram (20 by default).

GmmInit([*nc*,[*method*,[*means*]]]) Initialises the GMM parameters. *nc*: the number of Gaussian pdfs, *method*: type of parameter initialisation method, currently 1,2,3,4 are available. 1 is the simples one. *means*: if provided, this values are used as the mean vectors of the Gaussian pdfs. The means should be of a column vector, ie. each element is separated with ";".
e.g. GmmInit(2, 3, [0.5; 3.2]) will initialise a GMM with two Gaussian components, using the method 3, setting the mean values of the two Gaussian distributions to 0.5 and 3.2, respectively.

GmmTrain([*niter*]) Carries out the EM training algorithm iteratively. *niter*: the maximum number of iterations (100 by default). This function can be called more than once: the successive call starts with the value obtained by the last call of the function.

GmmPdf([*x*]) Calculates probability density of "x" using GMM. "x" can be a one data or a sequence of data (vector or matrix). To calculate an average likelihood for the data sequence "xn", do as follows

```
octave:> sum( GmmPdf(xn) ) / length(xn)
```

1.7 Variables

xn	: (vector)	training data. xn(1)∼xn(length(xn))	
xd	: (structure)	parameters for generating data using a GMM(gen)	
	xd.fname	GMM parameter file name	
	xd.m	means (vector)	
	xd.s	standard deviations (vector)	
	xd.w	mixture weights (vector)	
gmm	: (structure)	parameters and variables in GMM	
	gmm.nc	number of Gaussian pdf components	
	gmm.dim	dimensionality of vector space	
	gmm.init_method	type of methods for GMM parameter initialisation	
	gmm.Q	the most recent Q value .	
	gmm.Qk	Q value at the k-th iteration.	
	gmm.g	GMM parameters	
	gmm.w	GMM weight coefficients.	
gc	: (structure)	parameters for control graphics	
	gc.plot_true_pdf	Suppress drawing a true pdf in PlotHisPdf when it is 0	
	gc.hold_hist_axes	hold axes in PlotHist when it is true.	
	gc.sleep	sleep-time (in seconds) added at each iteration in GmmTrain() in order to make graph up	

* details should be found in "gmm-demo.m" and "gmm-em-2.m".

2 Exercises

2.1 Pdf estimation

- Investigate GMM training performance in terms of following factors:
  - size of data for training
  - the number of Gaussian components in GMM
  - initial values of GMM in the EM algorithm (You can try three types of initialisation in GmmInit())
  - the number of iteration in the EM algorithm

Supplemental information

- One indicator for performance comparison would be the (log) likelihood, which can be calculated using GmmPdf()
- You can try either of both type of data, i.e. the synthesised data by GMM using GenData() or real speech data.

2.2 Example for the EM algorithm

Find a good example in real world, in which the EM algorithm is needed for training, and describe the EM algorithm for the example you chose. Details of mathematical formulation are not needed, but clear definition of an optimisation problem should be given (i.e. what variables are observable and unobservable, what objective function is maximised with respect to which parameters?)